A Polynomial-Time Algorithm for Checking Equivalence Under Certain Semiring Congruences motivated by the State-space Isomorphism Problem for Hybrid Systems*

Bhaskar DasGupta†
Department of Computer Science
Rutgers University
Camden, NJ 08102, USA
Email: bhaskar@crab.rutgers.edu

Eduardo D. Sontag‡
Department of Mathematics
Rutgers University
New Brunswick, NJ 08903, USA
Email: sontag@control.rutgers.edu

Abstract

This paper presents a polynomial-time algorithm for equivalence under certain semiring congruences. These congruences arise when studying the isomorphism of state spaces for a class of hybrid systems. The area of hybrid systems concerns issues of modeling, computation, and control for systems which combine discrete and continuous components. The subclass of piecewise linear (PL) systems provides one systematic approach to discrete-time hybrid systems, naturally blending switching mechanisms with classical linear components. PL systems model arbitrary interconnections of finite automata and linear systems. Tools from automata theory, logic, and related areas of computer science and finite mathematics are used in the study of PL systems, in conjunction with linear algebra techniques, all in the context of a "PL algebra" formalism. PL systems are of interest as controllers as well as identification models.

Basic questions for any class of systems are those of equivalence, and, in particular, whether state spaces are equivalent under a change of variables. This paper studies this state-space equivalence problem for PL systems. The problem was known to be decidable, but its computational complexity was potentially exponential; here it is shown to be solvable in polynomial-time.

Keywords: Hybrid systems, Piecewise-linear systems, State-space equivalence, Semiring congruences, Polynomial time algorithms.

*An Extended Abstract of these results without most proofs appeared under the title *A Polynomial-Time Algorithm for an Equivalence Problem which Arises in Hybrid Systems Theory* in proceedings of the 37th IEEE Conference on Decision and Control, December 1998, IEEE Publications, pp. 1629-1634

†Supported in part by US Air Force Grant AFOSR-97-0159 and by NSF Grant CCR-9800086

‡Supported in part by US Air Force Grant AFOSR-97-0159

1 Introduction

Let $\mathcal{S} = \mathbb{N}[x, y]$ denote the collection of all polynomials in two commuting variables with *non-negative* integer coefficients. This set can be seen as a semiring, that is, sums and products obey all the usual rules, except that elements do not have additive inverses, i.e. one cannot "subtract".

Let $\sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ denote the following set of three equalities:

$$\begin{array}{rcl} \sigma_1 & : & x = 2x + 1 \\ \sigma_2 & : & y^2 = 2y^2 + y \\ \sigma_3 & : & y = x + y + 1 \end{array}$$

Let α_i and β_i denote the left-side and right-side, respectively, of the equality $\sigma_i \in \sigma$ (i.e., α_1 is x, β_1 is $2x + 1$, etc.). We let $=_\sigma$ be the semiring congruence generated by these equalities. Explicitly, this is defined as follows.

Definition 1.1 *Given two polynomials $P(x, y)$ and $Q(x, y)$ in $\mathcal{S}$, of same degree n, we say that $P(x, y)$ is equivalent to $Q(x, y)$ modulo the equalities in σ, or just that P is equivalent to Q modulo σ, provided that there is some sequence of polynomials*

$$P(x, y) \,=\, R_0(x, y)\,,\, R_1(x, y)\,,\, R_2(x, y)\,,\, \ldots,\, R_t(x, y) \,=\, Q(x, y)\,,$$

with each $R_i(x, y) \in \mathcal{S}$, such that for every $i > 0$, there are decompositions

$$R_i(x, y) \,=\, A(x, y)C(x, y) + T(x, y) \;\text{ and }\; R_{i-1}(x, y) = B(x, y)C(x, y) + T(x, y)\,,$$

where $C(x, y), T(x, y) \in \mathcal{S}$, with the property that $B(x, y)$ is obtained from $A(x, y)$ by applying one of the equalities in σ in either direction (i.e., either $B(x, y) = \alpha_i$ and $A(x, y) = \beta_i$ for some i, or $B(x, y) = \beta_i$ and $A(x, y) = \alpha_i$ for some i).

We use the notation $P(x, y) =_\sigma Q(x, y)$ to denote two polynomials equivalent as per Definition 1.1, and let $\neq_\sigma$ denote the negation of the relation $=_\sigma$.

Example: $2x^2 + 2y^2 + y =_\sigma 5x^2 + 2y^2 + 4x + y + 1$, since

$$2x^2 + 2y^2 + y = x^2 + x^2 + 2y^2 + y =_\sigma x^2 + (2x + 1)^2 + 2y^2 + y =_\sigma 5x^2 + 2y^2 + 4x + y + 1\,.$$

On the other hand, clearly $y^2 + y \neq_\sigma x^2$.

Notice that $P(x, y) =_\sigma Q(x, y)$ if and only if $Q(x, y) =_\sigma P(x, y)$, since each equality in σ can be applied in either direction, and hence $=_\sigma$ is an equivalence relation on $\mathcal{S}$.

An alternative way of stating the equivalence in Definition 1.1 is directly from the definition of semiring congruence and is as follows:

Definition 1.2 (Equivalent to Definition 1.1) *Let $\simeq$ be the smallest equivalence relation so that:*

- $A(x, y) \simeq A(x, y)$ *for all* $A(x, y) \in \mathcal{S}$
- $x \simeq 2x + 1$
- $y^2 \simeq 2y^2 + y$

- $y \simeq x + y + 1$
- $A(x,y)B(x,y) \simeq C(x,y)D(x,y)$ *if* $A(x,y) \simeq C(s,y)$ and $B(x,y) \simeq D(x,y)$.
- $A(x,y) + B(x,y) \simeq C(x,y) + D(x,y)$ *if* $A(x,y) \simeq C(s,y)$ and $B(x,y) \simeq D(x,y)$.

Then, P is equivalent to Q modulo σ if and only if P and Q are in the same equivalence class of the equivalence relation $\simeq$.

Our goal is to develop an efficient algorithm for deciding the following question: *Given P and Q, is $P =_\sigma Q$?*. It was shown in [17] that this question is algorithmically decidable. Two alternative proofs were sketched there, which were based, respectively, on the algorithms given for related semiring problems in the papers [6, 8]. However, no efficient algorithm was known for this procedure, nor was there a proof that the problem is intrinsically hard, i.e. NP-hard. *The purpose of this paper is to give a polynomial-time algorithm to decide if $P(x,y) =_\sigma Q(x,y)$.* Moreover, we provide an equivalence test based on the existence of canonical forms for elements on each class, which should be of independent interest.

Regarding the precise meaning of "polynomial-time computation," one should recall here that this term can be understood in two different ways. The first, the *unit cost model*, is intended to capture the algebraic complexity of a problem [5]; in that model, each arithmetic and comparison operation on two real numbers is assumed to take unit time. An alternative, the *logarithmic cost model*, is closer to the notion of computation in the usual Turing machine sense (e.g, see [7]); in this case one assumes that each coefficient of the two given polynomials is an integer with at most B bits, each arithmetic and comparison operation on two B bit integers takes $O(B)$ time, and the time involved in the decision procedure is required to be polynomial on B as well. For convenience, we will just write $O(\alpha)$ time to denote a running time of $O(\alpha)$ in the unit-cost model or a running time of $O(\alpha B)$ in the logarithmic-cost model (where B is the maximum number of bits needed to represent any integer involved in the arithmetic operations in the logarithmic-cost model).

The main theorem of this paper is the following.

Main Theorem. Given any two polynomials $P, Q \in \mathcal{S}$, each of degree at most n, whether $P =_\sigma Q$ or not can be decided in $O(n^2)$ time. Moreover, if $P =_\sigma Q$, then a sequence of applications of the equalities from σ (in either direction) transforming P to Q can be computed in $O(n^2)$ time.

The proof of the main theorem extends over the following few sections. Notice that the time taken by our algorithm for the unit-cost model is optimal within a constant factor in the worst case, since the polynomials have $O(n^2)$ coefficients. Similarly, the time for the logarithmic-cost model is also optimal within a constant factor in the worst case.

We assume that both given polynomials have the same degree, since application of any equality in σ preserves the degree of the polynomial. Unless otherwise stated explicitly, the following notations and conventions will be used throughout the rest of the paper. We will omit the indices in any summation $\sum_{i,j\geq 0\,;\, i+j\leq n}$, and denote such a sum simply by $\sum$. In general, $P(x,y) = \sum p_{i,j}x^i y^j \in \mathcal{S}$ and $Q(x,y) = \sum q_{i,j}x^i y^j \in \mathcal{S}$ will denote the two degree n polynomials involved in the decision procedure (with the condition that there exists some i', i'', j', j'', $i' + j' = i'' + j'' = n$ such that $p_{i',j'}, q_{i'',j''} > 0$). We also assume, without loss of generality, that $n \geq 1$; otherwise one cannot apply *any* rule from σ to a given polynomial of degree 0, and hence two degree 0 polynomials are transformable to each other if and only if they are *identical*. Sometimes we will omit the variables for clarity and simply refer to the two polynomials as P and Q. For the logarithmic cost model, we will also assume that each $p_{i,j}$ and $q_{i,j}$ is at most B bits long for some $B > 0$.

The following conventions are used throughout the paper in writing the algorithm for the proof of the Main Theorem. We write the algorithms in pseudo-code (which can be directly implemented in a standard programming language such as C or PASCAL). Comments are enclosed between /* and */ to explain the rationale behind various steps. A for loop of the form:

$$\begin{array}{l}\text{for } i = x, x+1, \ldots, y \text{ do} \\ \quad \vdots \\ \text{endfor}\end{array}$$

gets executed $y - x + 1$ times (for $i = x, x+1, x+2, \ldots, y$, in that order) if $y \geq x$, and **does not** get executed at all if $y < x$ (this is the same way for loops work in a language such as C). Similarly, a for loop of the form:

$$\begin{array}{l}\text{for } i = x, x-1, \ldots, y \text{ do} \\ \quad \vdots \\ \text{endfor}\end{array}$$

gets executed $y - x + 1$ times (for $i = x, x-1, x-2, \ldots, y$, in that order) if $y \leq x$, and **does not** get executed at all if $y > x$.

The rest of the paper is organized as follows. In Section 2 we discuss the general connection of the results in this paper with the area of hybrid systems; this entire section contains no new material, and is included for motivational reasons only. The class of PL systems is defined in the Section 2.1, and it is explained why it is precisely the problem mentioned in Section 1 that needs to be solved in order to solve a state-space equivalence problem. In Section 3 we discuss some preliminary results and show that an alternative way of looking at the given problem is via valid operations. In Section 4, we discuss some techniques to adjust the coefficients via valid operations. In Section 5, we give a necessary and sufficient condition under which a polynomial can be transformed to another via valid operations. In Section 6, we provide solution for the transformability problem for the simpler case when both the polynomials are in one variable x. Section 7 discusses the general case of two-variable polynomials and proves the main theorem. We conclude in Section 8 with some closing remarks.

2 Motivations from Hybrid Systems

The research works reported in this paper lie in the area of hybrid systems, which concerns itself with issues of modeling, computation and control for systems which combine discrete and continuous components. There are two main motivations for the study of hybrid systems. The first is the need to model and control devices whose description includes logical as well as continuous variables, as found routinely in consumer electronics. The second is the need for switching mechanisms when controlling even simple purely continuous systems (see for instance [19], Section 5.9). These motivations have given rise to a large amount of research dealing with different aspects of hybrid theory. See, for instance, the many papers in the volumes [1] and [12], and papers such as [4] for current control research.

This paper deals exclusively with computational complexity questions related to hybrid systems, in the style of the papers [2, 3, 9, 10, 11, 13, 14], and more specifically, in the context of the class of discrete-time PL systems.

In [16], the second author introduced an approach to hybrid systems modeling, via the class of *piecewise linear (PL) systems*. This class of systems allows the blending of switching mechanisms with classical linear components, and models arbitrary interconnections of finite automata and linear systems. Tools from automata theory, logic, and related areas of computer science and finite mathematics are used in the study of PL systems, in conjunction with linear algebra techniques, and combined into the logic formalism of the "PL algebra" introduced and developed in [17]. Besides being mathematically natural, being the smallest class which is closed under interconnections and which contains both linear systems and finite automata, PL systems may be used as identification models (by means of piecewise linear, e.g. linear spline, approximations), or as controllers for more general systems (the paper [16] established the theoretical possibility of stabilizing rather arbitrary systems using PL controllers in sample-and-hold mode).

Among the most basic questions which can be asked about any class of systems are those regarding equivalence, such as: given two systems, do they represent the same dynamics under a change of variables? As a preliminary step in answering such a question, one must determine if the state spaces of both systems are isomorphic in an appropriate sense. That is, one needs to know if an invertible change of variables is at all possible. Only later can one ask if the equations are the same. For classical, finite dimensional linear systems, this question is trivial, since only dimensions must match. For finite automata, similarly, the question is also trivial, because the cardinality of the state set is the only property that determines the existence of a relabeling of variables. For other classes of systems, however, the question is not as trivial, and single numbers such as dimensions or cardinalities may not suffice to settle the equivalence problem. For example, if one is dealing with continuous time systems defined by smooth vector fields on manifolds, the natural changes of variables are smooth transformations, and thus a system whose state space is a unit circle cannot be equivalent to a system whose state space is the real line, even though both systems have "dimension" one. Another illustration, more relevant to the present paper, is provided by systems whose variables are required to remain bounded, for instance, because of saturation effects; a state-space like the unit interval $[-1, 1]$ looks very different from the unbounded state-space $\mathbb{R}$, even though both have dimension one.

In this paper, we study this state-space equivalence problem for the PL systems studied in [16], under PL changes of variables. The main result provides a polynomial-time algorithm for the checking of PL isomorphism, assuming that the state space is represented in the specific normal form ("labels") introduced in [17]. Briefly, the paper [17] showed the decidability (recursive computability) of the equivalence problem, but the algorithm that would result from the discussion given there has in principle exponential time complexity (cf. [18] for some questions of quantifier elimination raised by that work). Obviously, having a polynomial time algorithm should have a major impact on future studies of PL systems.

2.1 PL Systems

Piecewise linear (or more precisely, piecewise-affine) systems, in the sense defined in [16], are discrete-time systems described by equations $x(t+1) = P(x(t), u(t))$ (we write simply "$x^+ = P(x, u)$") for which the transition mapping P is a PL map, that is, there is a decomposition of the state space $\mathcal{X}$ and the input value set $\mathcal{U}$ into finitely many pieces, such that, in each of these pieces, the mapping P is given by an affine function. The decomposition is required to be polyhedral, meaning that each piece is described by a set of linear equalities and inequalities.

For example, linear systems arise in the particular case in which there is just one region. But the PL system paradigm includes many more situations of interest, such as, to take just a few examples,

linear systems $x^+ = Ax + B\text{sat}\,(u)$ ($\text{sat}\,(u_1, \ldots, u_m)$ is the vector whose ith component is u_i if $|u_i| \leq 1$ and $\text{sign}\,(u_i)$ otherwise) whose actuators are subject to saturation, switched systems $x^+ = A_i x + B_i u$, where the choice of matrix pair (A_i, B_i) depends on a set of linear constraints on current inputs and states, or systems $x^+ = \text{sat}\,(Ax + Bu)$ for which underflows and overflows in state variables must be taken into account.

As part of the specification of a PL system, one includes explicit constraints on controls and states. Thus, the state space and control-value sets are taken to be subsets $\mathcal{X}$ and $\mathcal{U}$ of $\mathbb{R}^n$ and $\mathbb{R}^m$ respectively, which indicate *a priori* restrictions on the allowed ranges of variables. To make the theory stay "piecewise linear", we ask that these sets be definable in terms of a finite number of linear equalities and inequalities. Finite sets are included (n independent linear equalities specify each point), and in this way all finite automata are included as well.

Arbitrary interconnections of linear systems and finite automata can be modeled by PL systems, and vice-versa. More precisely, given any finite automaton with state space Q, input-value space T, and transition function $\delta : Q \times T \to Q$, we allow the state q of the automaton to control switching among $|Q|$ possible linear dynamics:

$$\begin{aligned} x^+ &= A_q x + B_q u + c_q \\ q^+ &= \delta(q, h(x, u)) \end{aligned}$$

where $A_1, \ldots, A_{|Q|}$ are matrices of size $n \times n$, $B_1, \ldots, B_{|Q|}$ are matrices of size $n \times m$, and $c_1, \ldots, c_{|Q|}$ are n-vectors, and where $h : \mathbb{R}^n \times \mathbb{R}^m \to T$ is a PL map (representing quantized observations of the linear systems). For a proof of this equivalence between PL systems and interconnections, as well as for precise definitions of PL systems and more discussion and examples, see the second author's paper in [1].

A more mathematically elegant definition of PL sets and maps can be given, as follows. The *PL subsets* of $\mathbb{R}^n$ are those belonging to the smallest Boolean algebra that contains all the open halfspaces of $\mathbb{R}^n$. A map $f : X \to Y$ between two PL subsets X and Y of $\mathbb{R}^a$ and $\mathbb{R}^b$ respectively, is a *PL map* if its graph is a PL subset of $\mathbb{R}^a \times \mathbb{R}^b$. By a *PL set* one means a PL subset of some $\mathbb{R}^n$. Finally, a PL system is a discrete-time system $x^+ = P(x, u)$ with PL state and input value sets and PL transition P.

Two PL sets X and Y are PL isomorphic if there are PL maps $f : X \to Y$ and $g : Y \to X$ such that $f \circ g$ and $g \circ f$ both equal the identity, that is, $y = f(x)$ is a bijective piecewise linear map.

A PL isomorphism is nothing else than an operation of the following type: make a finite number of cuts along a set of lines (or segments), apply an affine (linear plus translation) transformation to each piece (not dropping any lower-dimensional pieces), and finally paste it all together. As an example, let us take the interior of the triangle in $\mathbb{R}^2$ obtained as $\text{oc}\,\{(0,0),(1,1),(2,0)\}$, where we are using "oc" to indicate the interior of the convex hull of the corresponding points. (We can also define this set, of course, as the intersection of the three hyperplanes $x_2 > 0$, $x_1 - x_2 > 0$, and $x_1 + x_2 < 2$.) We now show that this triangle is PL isomorphic to the interior of the open square with vertices $(0,0)$, $(1,1)$, $(0,1)$, and $(1,0)$. First we cut along the segment $S_1 = \text{oc}\,\{(1,0),(1,1)\}$, obtaining the union of S_1, S_2, and S_3, where $S_2 = \text{oc}\,\{(0,0),(1,0),(1,1)\}$ and $S_3 = \text{oc}\,\{(1,1),(1,0),(2,0)\}$. Next, we apply the affine transformation

$$Tx = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} x - \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

to change S_3 into $S_3' = \text{oc}\,\{(1,1),(0,0),(0,1)\}$. Finally, we apply the affine transformation

$$Tx = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x - \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

to change S_1 into the missing diagonal $S_1' = \mathrm{oc}\,\{(0,0),(1,1)\}$, and we glue it all back. See Figure 1.

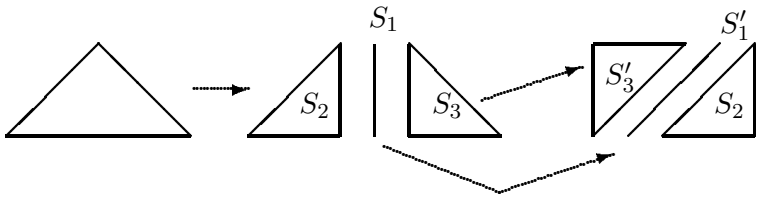

Figure 1: Example: triangle is PL isomorphic to square

One of the main results in the paper [17] provided a classification of PL sets under isomorphism. The critical step in this classification is to associate to each PL set X a "label" with the property that two spaces X and Y are isomorphic if and only if their labels are related in a certain manner. (By analogy, two finite-dimensional real vector spaces are linearly isomorphic if and only if their dimensions are the same, i.e., letting the "label" be the dimension, if their labels coincide. But in the PL case, single integers do not suffice as "labels".)

Labels are, by definition, polynomials in two variables x, y with non-negative integer coefficients. We let $\mathcal{S} = \mathbb{N}[x,y]$ denote the collection of all such polynomials. Examples of labels are 1, x, y, x^3, $1 + xy + x^2$, etc. We interpret the sum in $\mathcal{S}$ as union of disjoint sets and the product as Cartesian product of sets, the unit 1 as a one-element set, the variable x as the open interval $(0,1)$, and the variable y as the half-line $(0,+\infty)$. Thus, x^3 is an open cube, and $1 + xy + x^2$ is the union of a point, a disjoint set $(0,1)\times(0,+\infty)$, and a unit square disjoint from both. One may decompose any PL set into a finite union (algebraically, a sum) of objects each of which is linearly isomorphic to a monomial in x and y. (Simplicial decompositions provide a way to do this.) In this manner, a label (nonunique) can be associated to each PL set.

Certain formal equalities are easy to establish. Splitting the interval x as

$$(0,1) \;=\; (0,1/2) \bigcup \{1/2\} \bigcup (1/2,1)\,,$$

and then using affine maps ($t \mapsto 2t$ and $t \to 2t-1$ respectively) to map the first and last interval to x, we obtain "$x = 2x + 1$". On the other hand, the split $y = (0,+\infty) = (0,1)\bigcup\{1\}\bigcup(1,+\infty)$ (and $t \mapsto t-1$ applied to the last set) gives us the identity "$y = x + 1 + y$". Drawing a bisecting line through the first quadrant in $\mathbb{R}^2$ gives "$y^2 = y^2 + y + y^2$" (using, e.g., the linear transformation $(t_1,t_2) \mapsto (t_1 - t_2, t_2)$ to send the lower triangle $\{(t_1,t_2) | t_1 > 0, t_1 > t_2\}$ to y^2). See Figure 2. It was

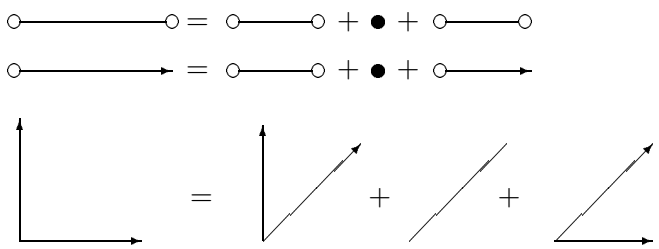

Figure 2: Elementary equivalences

shown in [17] that these three identities are enough, in the sense that two sets are isomorphic if and only if their labels can be obtained from each other by using repeatedly these elementary identities. In this paper, we take this result as a starting point.

3 Some Preliminary Results

Let $R = \sum r_{i,j}x^i y^j$ be an intermediate polynomial in a transformation from P to Q (initially, $R = P$).

Definition 3.1 *A monomial application of an equality from σ (in either direction) is an application of an equality of the type $C\alpha_i = C\beta_i$ (in either direction) for some i where C is a monomial (i.e., $C = x^j y^k$ for some indices j and k).*

It is easy to see that, without any loss of generality, it is sufficient to consider *only* monomial application of equalities from σ in any transformation of P to Q. Indeed, consider any *non-monomial* application of equalities from σ in the form $C\alpha_i = C\beta_i$ where C is *not* a monomial. Let $C = C_1 + C_2 + \cdots + C_\ell$ where $C_1, C_2, \ldots, C_\ell$ are monomials. Then, this is equivalent to ℓ monomial applications of equalities from σ of the form: $C_1\alpha_i = C_1\beta_i$, $C_2\alpha_i = C_2\beta_i$, ..., $C_\ell\alpha_i = C_\ell\beta_i$.

Hence, from now onwards we will consider **only monomial applications** of equalities from σ in our subsequent discussions whenever we apply an equality from σ in either direction.

Definition 3.2 *An elementary application of an equality from σ is one application of an equality from σ (in either direction) that changes two coefficients of R by exactly 1. A non-elementary application of an equality from σ is one application of an equality from σ which is not an elementary application.*

Observation 3.1 One elementary application *of an equality from σ changes the coefficients of R in one of the following manners:*

(a) *Substituting x by $2x + 1$ (forward direction of rule σ_1).*

Assume that the substitution involves the term $r_{i,j}x^i y^j$ (with $i, r_{i,j} \neq 0$) and is of the form:

$$r_{i,j}x^i y^j =_\sigma (r_{i,j} - 1)x^i y^j + x^{i-1}y^j(2x + 1) = (r_{i,j} + 1)x^i y^j + x^{i-1}y^j$$

This is equivalent to increasing both $r_{i,j}$ and $r_{i-1,j}$ by 1, as long as $i, r_{i,j} > 0$.

(b) *Substituting $2x + 1$ by x (backward direction of rule σ_1).*

Assume that the substitution involves the terms $r_{i,j}x^i y^j$ and $r_{i-1,j}x^{i-1}y^j$ and is of the form:

$$\begin{array}{rcl} r_{i,j}x^i y^j + r_{i-1,j}x^{i-1}y^j & = & (r_{i,j} - 2)x^i y^j + (r_{i-1,j} - 1)x^{i-1}y^j + x^{i-1}y^j(2x + 1) \\ & =_\sigma & (r_{i,j} - 2)x^i y^j + (r_{i-1,j} - 1)x^{i-1}y^j + x^i y^j \\ & = & (r_{i,j} - 1)x^i y^j + (r_{i-1,j} - 1)x^{i-1}y^j \end{array}$$

This is equivalent to decreasing both $r_{i,j}$ and $r_{i-1,j}$ by 1, as long as $r_{i,j} > 1$, $r_{i-1,j} > 0$, and $i > 0$.

(c) *Substituting y^2 by $2y^2 + y$ (forward direction of rule σ_2).*

By a similar reasoning as in (a), this is equivalent to increasing both $r_{i,j}$ and $r_{i,j-1}$ by 1, as long as $j > 1$ and $r_{i,j} > 0$.

(d) *Substituting $2y^2 + y$ by y^2 (backward direction of rule σ_2).*

By a similar reasoning as in (b), this is equivalent to decreasing both $r_{i,j}$ and $r_{i,j-1}$ by 1, as long as $j > 1$, $r_{i,j} > 1$ and $r_{i,j-1} > 0$.

(e) *Substituting y by $x+y+1$ (forward direction of rule σ_3).*

By a similar reasoning as in (a), this is equivalent to increasing both $r_{i,j-1}$ and $r_{i+1,j-1}$ by 1, as long as $j, r_{i,j} > 0$ and $i < n$.

(f) *Substituting $x+y+1$ by y (backward direction of rule σ_3).*

By a similar reasoning as in (b), this is equivalent to decreasing both $r_{i,j-1}$ and $r_{i+1,j-1}$ by 1, as long as $j, r_{i,j}, r_{i,j-1}, r_{i+1,j-1} > 0$ and $i < n$.

Lemma 3.1 *A non-elementary application of an equality from σ can be simulated by a sequence of elementary applications of the same equality from σ.*

Proof: We show the simulation for the case when $2x+1$ is substituted by x; the other cases are analogous. Consider the following non-elementary application of the equality (for some $c, i > 0$, $r_{i,j} \geq 2c$, $r_{i-1,j} \geq c$):

$$\begin{aligned} r_{i,j}x^iy^j + r_{i-1,j}x^{i-1}y^j &= (r_{i,j}-2c)x^iy^j + (r_{i-1,j}-c)x^{i-1}y^j + cx^{i-1}y^j(2x+1) \\ &=_\sigma (r_{i,j}-2c)x^iy^j + (r_{i-1,j}-c)x^{i-1}y^j + cx^iy^j \\ &= (r_{i,j}-c)x^iy^j + (r_{i-1,j}-c)x^{i-1}y^j\,. \end{aligned}$$

This can be simulated by c elementary applications of the same equality in the following manner:

$$\begin{aligned} r_{i,j}x^iy^j + r_{i-1,j}x^{i-1}y^j &=_\sigma (r_{i,j}-1)x^iy^j + (r_{i-1,j}-1)x^{i-1}y^j \quad \text{(first elementary application)} \\ &=_\sigma (r_{i,j}-2)x^iy^j + (r_{i-1,j}-2)x^{i-1}y^j \quad \text{(second elementary application)} \\ &\;\;\vdots \\ &=_\sigma (r_{i,j}-c)x^iy^j + (r_{i-1,j}-c)x^{i-1}y^j \quad \text{(last elementary application).} \end{aligned}$$

(Observe that only $r_{i,j} > c$ and $r_{i-1,j} > c-1$ are required when proceeding in this way, applying the transformations repeatedly.) ■

By Lemma 3.1, it is sufficient to restrict our attention only to sequences of *elementary (monomial) applications* of the equalities from σ.

For the purpose of proofs and descriptions of algorithms, it will be extremely convenient to describe the total effect of a sequence of $c > 0$ elementary applications of the same equality from σ on the same coefficients.

Repeating Observation 3.1 c times on the same coefficients gives rise to the following observation.

Observation 3.2 *A sequence of $c > 0$* **elementary application** *of the same equality from σ on the same coefficients changes these coefficients of R in one of the following manners:*

(a) *Substituting x by $2x+1$ (forward direction of rule σ_1).*

This is equivalent to increasing both $r_{i,j}$ and $r_{i-1,j}$ by c, as long as $i, r_{i,j} > 0$.

(b) *Substituting $2x+1$ by x (backward direction of rule σ_1).*

This is equivalent to decreasing $r_{i,j}$ and $r_{i-1,j}$ by c as long as $r_{i,j} > c$, $r_{i-1,j} > c-1$ and $i > 0$.

(c) *Substituting y^2 by $2y^2+y$ (forward direction of rule σ_2).*

This is equivalent to increasing both $r_{i,j}$ and $r_{i,j-1}$ by c, as long as $j > 1$ and $r_{i,j} > 0$.

(d) *Substituting $2y^2 + y$ by y^2 (backward direction of rule σ_2).*

This is equivalent to decreasing both $r_{i,j}$ and $r_{i,j-1}$ by c, as long as $j > 1$, $r_{i,j} > c$ and $r_{i,j-1} > c - 1$.

(e) *Substituting y by $x + y + 1$ (forward direction of rule σ_3).*

This is equivalent to increasing both $r_{i,j-1}$ and $r_{i+1,j-1}$ by c, as long as $j, r_{i,j} > 0$ and $i < n$.

(f) *Substituting $x + y + 1$ by y (backward direction of rule σ_3).*

This is equivalent to decreasing both $r_{i,j-1}$ and $r_{i+1,j-1}$ by c, as long as $j, r_{i,j} > 0$, $r_{i,j-1}, r_{i+1,j-1} \geq c$ and $i < n$.

Definition 3.3 *A* **valid operation** *on the coefficients of a polynomial $R \in \mathcal{S}$ is one of the following operations:*

- *For some $r_{i,j} > 0$, do one of the following:*

  Move I(a): *if $i > 0$, increase both $r_{i,j}$ and $r_{i-1,j}$ by an arbitrary integer $c > 0$.*

  Move I(b): *if $j > 1$, increase both $r_{i,j}$ and $r_{i,j-1}$ by an arbitrary integer $c > 0$.*

  Move I(c): *if $j > 0$, increase both $r_{i,j-1}$ and $r_{i+1,j-1}$ by an arbitrary integer $c > 0$.*

  Move I(d): *if $j > 0$, $r_{i,j-1}, r_{i+1,j-1} \geq c$ for some arbitrary integer $c > 0$, decrease both $r_{i,j-1}$ and $r_{i+1,j-1}$ by c.*

- *For some $r_{i,j} > c > 0$, do one of the following:*

  Move II(a): *if $i > 0$ and $r_{i-1,j} \geq c$, decrease both $r_{i,j}$ and $r_{i-1,j}$ by c.*

  Move II(b): *if $j > 1$ and $r_{i,j-1} \geq c$, decrease both $r_{i,j}$ and $r_{i,j-1}$ by c.*

Note that these rules correspond, respectively, to (a), (c), (e), (f), (b), and (d) in Observation 3.2.

Hence, the problem to decide if $P =_\sigma Q$ can be alternatively formulated in terms of valid operations as follows.

INSTANCE: Two polynomials $P, Q \in \mathcal{S}$.

QUESTION: Is there a sequence of zero or more valid operations that changes the coefficients of P such that at the end $p_{i,j} = q_{i,j}$ for all i and j?

Definition 3.4 *For a coefficient $r_{i,j}$ of a polynomial $R \in \mathcal{S}$, the* **neighbors** *of $r_{i,j}$, denoted by $N(r_{i,j})$, is the set of (at most 4) coefficients $r_{i',j'}$ such that $|i - i'| + |j - j'| = 1$.*

We will, from now on, interchangeably look at either the original problem or at the above formulation of the problem through valid operations, whichever is more convenient for our purpose.

4 Some Techniques On Modifying Coefficients

In this section, we discuss some results and techniques to modify the coefficients $r_{i,j}$ of $R \in \mathcal{S}$ by *valid operations*, where R is assumed to be a polynomial of degree n obtained by performing zero or more valid operations on P.

Lemma 4.1 *Let $r_{i,j} > 0$ for some indices i and j, $K > 0$ be an arbitrary positive integer and $L < r_{i,j}$ be an arbitrary non-negative integer. Then, there is a sequence of at most $O(n)$ valid operations changing the coefficients of R such that the final values of the coefficients, $r'_{i,j}$, satisfy one or more of the following (coefficients not explicitly mentioned do not change their values):*

(a) *If $i > 0$, then $r'_{i',j} \geq K$ for all $i'' \leq i' < i$ for any $i'' \geq 0$.*

(b) *If $i > 0$, then $r'_{i,j} = r_{i,j} - L$ and $r'_{i'',j}$ is increased (if $i'' - i$ is even) or decreased (if $i'' - i$ is odd) by L from its old value $r_{i'',j}$, for any i'' satisfying $0 \leq i'' < i$ and $r_{i'',j} \geq L$.*

(c) *If $i > 0$, then $r'_{i,j} = r_{i,j} + K$ and $r'_{i'',j}$ is increased or decreased by K from its old value $r_{i'',j}$, for any i'' satisfying $0 \leq i'' < i$ and $r_{i'',j} \geq K$.*

Proof: **(a)** Since $r_{i,j} > 0$, we can use the following sequence of valid operations, each of type I(a): increase $r_{i,j}$ and $r_{i-1,j}$ by K; then, increase $r_{i-1,j}$ and $r_{i-2,j}$ by K; and continue this way until we have increased the pair $r_{i''+1,j}$ and $r_{i'',j}$ by K. Obviously, we need at most n valid operations.

(b) Assume $L > 0$, since otherwise there is nothing to do. Since $r_{i,j} > 0$, we can use the following sequence of valid operations:

(i) If $i'' < i - 1$, we use a sequence of at most n valid operations (each of type I(a)) as described in part (a) (with $K = L$) to ensure that $r_{i',j} > L$ for all $i'' < i' \leq i$. Notice that, if $i'' < i - 1$, then this step increases the values of the coefficients $r_{i-1,j}, r_{i-2,j}, \ldots, r_{i''-1,j}$ by $2L$ and increases $r_{i,j}$ by L.

(ii) Use the following sequence of (at most n) alternately decreasing (type II(a)) and increasing (type I(a)) valid operations: decrease $r_{i,j}$ and $r_{i-1,j}$ by L; next, increase $r_{i-1,j}$ and $r_{i-2,j}$ by L; next, decrease $r_{i-2,j}$ and $r_{i-3,j}$ by L; and so forth, until we have performed valid operation on the pair $r_{i''-1,j}$ and $r_{i'',j}$. Note that this step restores the old value of $r_{i,j}$, increases or decreases the value of $r_{i'',j}$ obtained after (i) by L, and leaves other coefficients unchanged.

(iii) Finally, if $i'' < i - 1$, we use the reverse sequence of valid operations as used in (i) to decrease by $2L$ the coefficients $r_{i',j}$ for $i'' < i' < i$ and decrease by L the coefficient $r_{i,j}$. Notice that, if $i'' < i-1$, this step decreases the value of $r_{i,j}$ obtained after (ii) by L (and, hence $r'_{i,j} = r_{i,j} - L$), and restores the initial values of the coefficients $r_{i-1,j}, r_{i-2,j}, \ldots, r_{i''-1,j}$ (that is, $r'_{i-1,j} = r_{i-1,j}$, $r'_{i-2,j} = r_{i-2,j}$, …, $r'_{i''-1,j} = r_{i''-1,j}$).

If $i'' = i' - 1$, then only step (ii) is performed, which produces the desired result.

(c) This is very similar to (b) above. We provide details of the steps below for the sake of completeness.

(i) If $i'' < i - 1$, we use a sequence of at most n valid operations (each of type I(a)) as described in part (a) to ensure that $r_{i',j} > K$ for all $i'' < i' \leq i$. Notice that, if $i'' < i - 1$, then this step increases the values of the coefficients $r_{i-1,j}, r_{i-2,j}, \ldots, r_{i''-1,j}$ by $2K$ and increases $r_{i,j}$ by K.

(ii) Use the following sequence of (at most n) alternately decreasing (type II(a)) and increasing (type I(a)) valid operations: increase $r_{i,j}$ and $r_{i-1,j}$ by K; next, decrease $r_{i-1,j}$ and $r_{i-2,j}$ by K; next, increase $r_{i-2,j}$ and $r_{i-3,j}$ by K; and so forth, until we have performed valid operation on the pair $r_{i''-1,j}$ and $r_{i'',j}$. Note that this step increases the value of $r_{i,j}$ obtained after (i) by K, increases or decreases the value of $r_{i'',j}$ obtained after (i) by K, and leaves other coefficients unchanged.

(iii) Finally, if $i'' < i-1$, we use the reverse sequence of valid operations as used in (i) to decrease by $2K$ the coefficients $r_{i',j}$ for $i'' < i' < i$ and decrease by K the coefficient $r_{i,j}$. Notice that, if $i'' < i-1$, this step decreases the value of $r_{i,j}$ obtained after (ii) by K (and, hence $r'_{i,j} = r_{i,j}+K$), and restores the initial values of the coefficients $r_{i-1,j}, r_{i-2,j}, \ldots, r_{i''-1,j}$ (that is, $r'_{i-1,j} = r_{i-1,j}$, $r'_{i-2,j} = r_{i-2,j}$, ..., $r'_{i''-1,j} = r_{i''-1,j}$).

If $i'' = i' - 1$, then only step (ii) is performed, which produces the desired result. ■

5 Necessary and Sufficient Conditions for Transformability (and Canonical Forms)

In this section, we discuss some necessary (and, sometimes sufficient) conditions for transforming the polynomial P to Q. First, we state an easy necessary condition.

Proposition 5.1 *$P =_\sigma Q$ implies $\sum_i \sum_j p_{i,j} = \sum_i \sum_j q_{i,j} \pmod 2$.*

Proof: Every valid operation on a polynomial increases or decreases the total sum of all the coefficients of the polynomial by an even integer. ■

The condition stated in Proposition 5.1 is obviously not sufficient, since, for example, if $P = y+1$ and $Q = 2y+2$, then $\sum_i \sum_j p_{i,j} = \sum_i \sum_j q_{i,j} \pmod 2$, but $P \neq_\sigma Q$. Hence, we need to strengthen the condition.

Definition 5.1 *Let $R = \sum r_{i,j} x^i y^j \in \mathcal{S}$ and n be the original degree of the given polynomial P. Define the* **characteristic function** *$\Lambda_n : R \mapsto \mathcal{N}$ to be the following function:*

$$\Lambda_n(R) = \sum_{i=0}^{n} \sum_{j=0}^{n-i} (-1)^{n-j-i} r_{i,j}$$

Figure 3 illustrates how $\Lambda_n(R)$ is calculated when $n = 5$. Its idea is motivated by the *Euler characteristic*[1] function of an abstract simplicial complex on a set (see [20]). Notice that the sign of every coefficient of R is *exactly the opposite* as that of any of its (at most 4) neighbors. The number $\Lambda_n(R)$ can be computed trivially in $O(n^2)$ time. Note that the Euler characteristic is intimately related to the definition of labels of PL sets and the relations σ; as a matter of fact, the paper [17] (page 200, formula (3.28) and following discussion) explains how the classical theorem on Euler characteristics of polyhedra is a simple consequence of PL set theory.

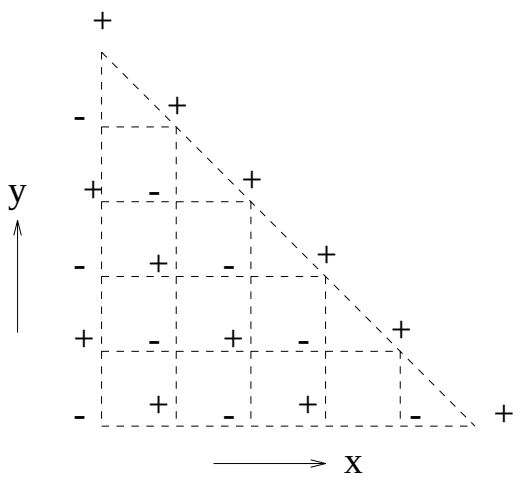

Figure 3: Illustration of how $\Lambda_n(R)$ is calculated (for $n = 5$) for a polynomial $R \in \mathbb{Z}[x, y]$. The grid-point with coordinates (i, j) stores the coefficient $r_{i,j}$. A "+" sign (resp., "−" sign) in the figure for the grid-point for $r_{i,j}$ indicates that this element is added (resp., subtracted) in the summation of $\Lambda_n(R)$.

Note that Λ_n is a linear operator. From now on, for notational convenience, we will drop the subscript n from Λ_n with the understanding that the subscript is always the degree of given polynomial P.

Lemma 5.1 *Let $P \in \mathcal{S}$. Let $P' \in \mathcal{S}$ be any polynomial obtained by applying a valid operation on P. Then, $\Lambda(P') = \Lambda(P)$.*

Proof: Definition 3.3 shows that a valid operation on the polynomial P either changes both $p_{i,j}$ and $p_{i-1,j}$ by the same amount or it changes both $p_{i,j}$ and $p_{i,j-1}$ by the same amount, for some appropriate indices i and j. In the former case, since $p_{i,j}$ and $p_{i-1,j}$ have opposite signs in the summation for $\Lambda(P)$, the value of $\Lambda(P)$ remains unchanged. In the later case also, since $p_{i,j}$ and $p_{i,j-1}$ have opposite signs in the summation for $\Lambda(P)$, the value of $\Lambda(P)$ remains unchanged. Hence, $\Lambda(P') = \Lambda(P)$. ■

Lemma 5.2 *Let $P, Q \in \mathcal{S}$. Then, $P =_\sigma Q$ implies $\Lambda(P) = \Lambda(Q)$.*

Proof: Suppose that $P =_\sigma Q$. We showed in Lemma 5.1 that if P is changed to P' by a single valid operation, then $\Lambda(P) = \Lambda(P')$. This proves (by induction on the length of the transformation sequence) that the value of $\Lambda(P)$ must be equal to $\Lambda(Q)$. ■

Corollary 5.1 *Let P and Q differ in exactly one coefficient. Then, $P \neq_\sigma Q$.*

Notice that $\Lambda(P) = \Lambda(Q)$ implies $\sum_i \sum_j p_{i,j} = \sum_i \sum_j q_{i,j} \pmod 2$. However, the conditions stated in Lemma 5.2 is still not sufficient. For example, again if $P = y + 1$ and $Q = 2y + 2$, then $\Lambda(P - Q) = 0$, but $P \neq_\sigma Q$. Hence, we need to further strengthen the condition in Lemma 5.2.

[1] If f_i is the number of faces of dimension i of an abstract simplicial complex Δ, then the Euler characteristic of Δ is $\sum_{i \geq 0} (-1)^i f_i$

Definition 5.2 *For any polynomial* $R = \sum r_{i,j}x^iy^j \in \mathcal{S}$, *define* $R^1 = \sum_{i=0}^{n} r_{i,0}x^i$ *and* $R^2 = R - R^1$ *(notice that* $R^1, R^2 \in \mathcal{S}$*).*

Now, we have the following lemma.

Lemma 5.3 *If* $P =_\sigma Q$, *then* $\Lambda(P^1) = \Lambda(Q^1)$ **and** $\Lambda(P^2) = \Lambda(Q^2)$.

Proof: There is no valid move that changes two coefficients of P one of which is in P^1 and the other in P^2. Hence, the result follows using a proof similar to that of Lemma 5.2 separately on the polynomial pairs P^1, Q^1 and P^2, Q^2. ■

Notice that, by linearity of the function Λ, $\Lambda(P^1) = \Lambda(Q^1)$ and $\Lambda(P^2) = \Lambda(Q^2)$ also implies $\Lambda(P) = \Lambda(Q)$. Unfortunately, the condition stated in Lemma 5.3 is still not a sufficient condition. For example, consider the two polynomials $P = xy^2 + x^5$ and $Q = x^3y^2 + x^5$, each of degree 5 ($n = 5$). Hence, $P^1 = Q^1 = x^5$, $P^2 = xy^2$, $Q^2 = x^3y^2$ and $\Lambda(P^1) = \Lambda(P^2) = \Lambda(Q^1) = \Lambda(Q^2) = 1$. But, it can be easily verified that $P \neq_\sigma Q$, since no sequences of valid operations on P can increase the coefficient $p_{3,2}$ to a non-zero value.

To develop a necessary as well as sufficient condition for transformability, we need to state a few more definitions.

Definition 5.3 *Given a polynomial* $P \in \mathcal{S}$, *the transitive closure of* P, *denoted by* $Trans(P)$, *is the (infinite) family of all polynomials in which each polynomial* $P' = \sum_{i,j} p'_{i,j}x^iy^j$ *in the family is obtained by applying zero or more valid operations on* P *such that the coefficients of* P' *satisfy the following condition: if* $p'_{i,j} = 0$ *for some indices* i *and* j, *then there is no sequence of zero or more valid operations that can increase* $p'_{i,j}$ *to a non-zero value.*

For example, if $P = y^2 + x^5$, then $y^2 + y + xy + 2x^5 + 2x^4 + x^3 + x^2 + 2x + 1$ is a member of $Trans(P)$. Although $Trans(P)$ is an infinite family of polynomials, for the purpose of this paper it suffices to compute just one member of $Trans(P)$ in polynomial time. The following is a high-level description of an algorithm to compute one member of $Trans(P)$ in $O(n^2)$ time.

```
done=FALSE
while (done=FALSE) do
    if there is a valid operation which can increase the value of some p_{i,j} from zero to one
        perform this valid operation
     else done=TRUE
endwhile
```

The above high-level description can be implemented to run in $O(n^2)$ time in the following way.

```
for j = n, n − 1, . . . , 0 do
    for i = n − j, n − j − 1, n − j − 2, . . . , 0 do

        if p_{i,j} > 0 then
            if i > 1 and p_{i−1,j} = 0 then increase p_{i,j} and p_{i−1,j} by 1 (move I(a))
            if j > 1 and p_{i,j−1} = 0 then increase p_{i,j} and p_{i,j−1} by 1 (move I(b))
            if j > 0 and i < n and p_{i+1,j−1} = 0 then increase p_{i,j−1} and p_{i+1,j−1} by 1 (move I(c))

    endfor
endfor
```

Definition 5.4 *The support set of a polynomial $P \in \mathcal{S}$, denoted by $Support(P)$, is defined as*

$$Support(P) \;=\; \{\, (i,j) \,|\, p_{i,j} > 0 \,\}$$

For example, if $P = y^2 + x^5$, then $Support(P)$ is the set $\{(5,0),(0,2)\}$. Notice that if P' and P'' are two members of $Trans(P)$, then $Support(P') = Support(P'')$. Now, we are ready to state the necessary and sufficient conditions for transformability.

Lemma 5.4 *$P =_\sigma Q$ if and only if all of the following conditions are satisfied:*

(a) *Degree of P is the same as degree of Q.*

(b) $\Lambda(P^1) = \Lambda(Q^1)$

(c) $\Lambda(P^2) = \Lambda(Q^2)$

(d) *$Support(P') = Support(Q')$, where P' and Q' are any members of $Trans(P)$ and $Trans(Q)$, respectively.*

(Notice that all these conditions can be checked in $O(n^2)$ time)

Lemma 5.4 in fact shows the following canonical forms for a given polynomial $P \in \mathcal{S}$ of degree $n > 0$.

Corollary 5.2 (Canonical forms) *Let $P \in \mathcal{S}$ be of degree $n > 0$ and let $P' = \sum p'_{i,j} x^i y^j$ be any member of $Trans(P)$. Let $0 \le j_1 \le n$ be the largest index in P' such that there exists some i with $p'_{i,j_1} > 0$. For each $0 \le j \le j_1$, let i_j be the largest index in P' such that $p'_{i_j,j} > 0$. Then, P can be transformed, in $O(n^2)$ time, to a unique polynomial of the form $A + B$, where the polynomial A is one of the following two forms:*

$$\Lambda(P^1)x^n, \quad \textit{if } \Lambda(P^1) > 0$$

$$x^n + (1 - \Lambda(P^1))x^{n-1}, \quad \textit{if } \Lambda(P^1) \le 0$$

and the polynomial B is one of the following six forms:

$$0, \quad \textit{if } j_1 = 0$$

$$p_{0,1}y, \quad \textit{if } j_1 = 1,\ i_1 = 0$$

$$|\Lambda(P^2)|x^{i_1}y, \quad \begin{array}{l}\textit{if } j_1 = 1,\ i_1 > 0,\ \Lambda(P^2) > 0,\ (-1)^{n-i_1-1} > 0 \\ \textit{or, if } j_1 = 1,\ i_1 > 0,\ \Lambda(P^2) < 0,\ (-1)^{n-i_1-1} < 0\end{array}$$

$$x^{i^1}y + (1 + |\Lambda(P^2)|)x^{i_1-1}y, \quad \begin{array}{l}\textit{if } j_1 = 1,\ i_1 > 0,\ \Lambda(P^2) > 0,\ (-1)^{n-i_1-1} < 0 \\ \textit{or, if } j_1 = 1,\ i_1 > 0,\ \Lambda(P^2) < 0,\ (-1)^{n-i_1-1} > 0 \\ \textit{or, if } j_1 = 1,\ i_1 > 0,\ \Lambda(P^2) = 0\end{array}$$

$$(\textstyle\sum_{j=1}^{j_1-1} x^{i_j}y^j) + (1 + |\Lambda(P^2) - \sum_{j=1}^{j_1}(-1)^{n-j-i_j}|)x^{i_{j_1}}y^{j_1}, \quad \begin{array}{l}\textit{if } j_1 > 1,\ \Lambda(P^2) \geq \sum_{j=1}^{j_1}(-1)^{n-j-i_j}, \\ \qquad \textit{and } (-1)^{n-j_1-i_{j_1}} > 0 \\ \textit{or, if } j_1 > 1,\ \Lambda(P^2) < \sum_{j=1}^{j_1}(-1)^{n-j-i_j}, \\ \qquad \textit{and } (-1)^{n-j_1-i_{j_1}} < 0\end{array}$$

$$(\textstyle\sum_{j=1}^{j_1} x^{i_j}y^j) + (\,|\Lambda(P^2) - \sum_{j=1}^{j_1}(-1)^{n-j-i_j}|\,)x^{i_{j_1}}y^{j_1-1}, \quad \begin{array}{l}\textit{if } j_1 > 1,\ \Lambda(P^2) \geq \sum_{j=1}^{j_1}(-1)^{n-j-i_j}, \\ \qquad \textit{and } (-1)^{n-j_1-i_{j_1}} < 0 \\ \textit{or, if } j_1 > 1,\ \Lambda(P^2) < \sum_{j=1}^{j_1}(-1)^{n-j-i_j}, \\ \qquad \textit{and } (-1)^{n-j_1-i_{j_1}} > 0\end{array}$$

Proof: It can checked that for each canonical form, the conditions (a), (b) (c) and (d) of Lemma 5.4 are satisfied. Moreover, given two different canonical forms, it can also be easily seen that at least one of the conditions (a), (b), (c) or (d) of Lemma 5.4 is violated. ■

Hence, to check if two polynomials are equal, one just needs to transform each of them to their respective canonical form (via Corollary 5.2) and then check if the two canonical forms are identical.

The "only if" part of Lemma 5.4 follows immediately from Lemma 5.3 and the following lemma.

Lemma 5.5 *If $Support(P') \neq Support(Q')$, then $P \neq_\sigma Q$.*

Proof: Assume $Support(P') \neq Support(Q')$ and let $X = Support(P') \oplus Support(Q') \neq \emptyset$, where $\oplus$ is the set symmetric difference operator. Let $(i, j) \in X$. Assume, without loss of generality, that $p'_{i,j} \in P'$ and $q'_{i,j} \in Q'$ are the two coefficients with $p'_{i,j} > 0$ and $q'_{i,j} = 0$. Since Q' is a member of $Trans(Q)$, $q'_{i,j}$ cannot be changed to a non-zero value by any sequence of valid operations. Hence, $P \neq_\sigma Q$. ■

The "if" part of Lemma 5.4 will follow by a careful inspection of the algorithm in the proof of the Main Theorem to be discussed in the next few sections.

6 The One-Variable Case

In this section we consider the simpler case when both P and Q are polynomials in only one variable x (i.e., $p_{i,j} = q_{i,j} = 0$ for all $j > 0$). This simpler case is important because a technique similar to the one employed for its solution will be applied repeatedly (with appropriate modifications) to solve the original problem. For notational convenience, let us denote $p_{j,0}$ (resp. $q_{j,0}$) simply by p_j

(resp. q_j). Notice that $p_n, q_n > 0$ (since both P and Q are of degree n) and $\Lambda(P^2) = \Lambda(Q^2) = 0$ (since $P^2 = Q^2 = 0$). Moreover, $Support(P') = Support(Q') = \{(0,0), (1,0), (2,0), \ldots, (n,0)\}$ for any $P' \in Trans(P)$ and $Q' \in Trans(Q)$.

Theorem 6.1 *In the one-variable case, $P =_\sigma Q$ if and only if $\Lambda(P^1) = \Lambda(Q^1)$. Moreover, if $P =_\sigma Q$, then a sequence of valid operations transforming P to Q can be computed in $O(n)$ time.*

Proof: It turns out that if $P =_\sigma Q$, then P can be transformed to Q by using only the equality σ_1. The proof will be constructive and will produce an $O(n)$ time algorithm to give a sequence of transformations from P to Q (or, report that $P \neq_\sigma Q$). The algorithm works in two steps:

Step 1: Transform P to another polynomial P' using the equality σ_1 such that P' differs from P in *at most* one coefficient.

Step 2: Use Corollary 5.1 and transitivity to conclude that $P =_\sigma Q$ if and only if P' is the same as Q.

Since Step 2 is trivial, we concentrate on Step 1. We assume $\Lambda(P^1) = \Lambda(Q^1)$ (since otherwise $P \neq_\sigma Q$). We specify how to modify $p_0, p_1, p_2, \ldots, p_{n-2}$, in that order, using valid operations corresponding to the equality σ_1, such that at the end we have $p_0 = q_0$, $p_1 = q_1$,$\ldots$,$p_{n-2} = q_{n-2}$. First, since $p_n > 0$, use Lemma 4.1 (part (a)) to increase $p_n, p_{n-1}, \ldots, p_0$ by at least 1. This takes $O(n)$ time. Notice that, after this step, $p_i > 0$ for all indices i and the value of $\Lambda(P^1)$ remains the same as before.

Inductively, assume that before the i^{th} step of coefficient-correction ($i = 0, 1, 2, \ldots, n-2$), we have already corrected the coefficients $p_0, p_1, \ldots, p_{i-1}$ (i.e., have ensured $p_0 = q_0$, $p_1 = q_1$, $\ldots$, $p_{i-1} = q_{i-1}$), possibly by modifying some of the coefficients $p_i, \ldots, p_n$, but **maintaining the following invariant**: $p_i, p_{i+1}, \ldots, p_n > 0$ before, during or after the correction step. We now show how to correct the current value of p_i (i.e., to ensure $p_i = q_i$ if $p_i \neq q_i$) and maintain the invariant that $p_{i+1}, p_{i+2}, \ldots, p_n > 0$. There are three cases to consider:

Case 1. $p_i < q_i$. Then, since $p_{i+1} > 0$, increase both p_i and p_{i+1} by $q_i - p_i$ (move I(a)).

Case 2. $p_i > q_i$ and $p_{i+1} > (p_i - q_i)$. Then, decrease both p_i and p_{i+1} by $p_i - q_i$ (move II(a)).

Case 3. $p_i > q_i$ and $p_{i+1} \leq (p_i - q_i)$. Then, (since $p_{i+2} > 0$) first increase both p_{i+1} and p_{i+2} by $(p_i - q_i) - p_{i+1} + 1$ (move I(a)). Then, decrease both p_i and p_{i+1} by $p_i - q_i$ (move II(a)).

Hence, after the $(n-2)^{th}$ step of coefficient-correction, we have changed the coefficients such that all but at most the leading two coefficients, p_n and p_{n-1}, differ from their corresponding coefficients q_n and q_{n-1} in the polynomial Q. Finally, since $\Lambda(P^1) = \Lambda(Q^1)$, and the value of $\Lambda(P^1)$ does not change by any sequence of valid operations, we must have $p_n - q_n = p_{n-1} - q_{n-1}$. Also, remember that $q_n > 0$ and $q_{n-1} \geq 0$. If $p_n < q_n$, we increase both p_{n-1} and p_n by $|p_n - q_n|$, otherwise, if $p_n > q_n$, we decrease p_{n-1} and p_n by $p_n - q_n$. This completes the transformation. Our complete algorithm is as shown in the next page.

It is clear that the total time taken by the above algorithm is $O(n)$. Notice also P can always be transformed to Q provided $\Lambda(P^1) = \Lambda(Q^1)$. This completes the proof. ■

The proof has actually shown, as well, the following interesting facts:

Corollary 6.1 *In the one-variable case, given any two polynomials P and Q of the same degree n, it is always possible to transform P in $O(n)$ time, using valid operations, to a polynomial P' which differs from Q in at most the two coefficients p_{n-1} and p_n.*

Corollary 6.2 (Special case of Corollary 5.2) *In the one-variable case, any polynomial $P \in \mathcal{S}$ of degree $n \geq 1$ can be transformed using $O(n)$ valid operations to the following polynomial:*

- $\Lambda(P)x^n$, *if* $\Lambda(P) > 0$.
- $x^n + (1 - \Lambda(P))x^{n-1}$, *if* $\Lambda(P) \leq 0$.

if $\Lambda(P^1) = \Lambda(Q^1)$ then

    for $i = n, n-1, n-2, \ldots, 1$ do
        increase p_i and p_{i-1} by 1 (move I(a))
    endfor

    for $i = 0, 1, 2, \ldots, n-2$ do
        if $p_i < q_i$, increase p_i and p_{i+1} by $q_i - p_i$ (move I(a))
        if $p_i > q_i$ **and** $p_{i+1} > (p_i - q_i)$, decrease p_i and p_{i+1} by $p_i - q_i$ (move II(a))
        if $p_i > q_i$ **and** $p_{i+1} \leq (p_i - q_i)$ then
            increase p_{i+1} and p_{i+2} by $(p_i - q_i) - p_{i+1} + 1$ (move I(a))
            decrease p_i and p_{i+1} by $p_i - q_i$ (move II(a))
        endif
    endfor

    if $p_n < q_n$, increase p_{n-1} and p_n by $|p_n - q_n|$ (move I(a))
    if $p_n > q_n$, decrease p_{n-1} and p_n by $p_n - q_n$ (move II(a))
    Now P must be identical to Q, hence report $P =_\sigma Q$ and exit

else $P \neq_\sigma Q$.
endif

Algorithm for the one-variable case (Section 6).

7 The Two-Variable Case (Original Problem)

Now, we consider the original problem when both of P and Q, each of the same degree n, have non-zero coefficients for terms involving positive powers of y (i.e., $P^2, Q^2 \neq 0$). The main result of this section is as follows.

Lemma 7.1 *Assume that all of the following conditions are satisfied:*

(a) *Degree of P is the same as degree of Q.*

(a) $\Lambda(P^1) = \Lambda(Q^1)$

(b) $\Lambda(P^2) = \Lambda(Q^2)$

(c) *$Support(P') = Support(Q')$, where P' and Q' are any members of $Trans(P)$ and $Trans(Q)$, respectively.*

Then, $P =_\sigma Q$. Moreover, a transformation sequence from P to Q can be computed in $O(n^2)$ time.

The rest of this section is devoted to a proof of Lemma 7.1. Notice that Lemma 7.1, together with Lemma 5.3 and Lemma 5.5, prove Lemma 5.4 and the main theorem.

Assume that conditions (a), (b), (c) and (d) of Lemma 7.1 hold. Notice that these conditions can be checked in $O(n^2)$ time. We will transform both P and Q, if necessary, until they become identical.

First, in Step 1, we replace P and Q by two members P' and Q' of $Trans(P)$ and $Trans(Q)$, respectively. From the discussion following Definition 5.3, this can be done in $O(n^2)$ time. The algorithm for this part was already given in Section 5 following Definition 5.3 and is repeated merely for completeness.

```
for j = n, n − 1, ..., 0 do
   for i = n − j, n − j − 1, n − j − 2, ..., 0 do

      if p_{i,j} > 0 then
         if i > 1 and p_{i−1,j} = 0 then increase p_{i,j} and p_{i−1,j} by 1 (move I(a))
         if j > 1 and p_{i,j−1} = 0 then increase p_{i,j} and p_{i,j−1} by 1 (move I(b))
         if j > 0 and i < n and p_{i+1,j−1} = 0 then increase p_{i,j−1} and p_{i+1,j−1} by 1 (move I(c))

      if q_{i,j} > 0 then
         if i > 1 and q_{i−1,j} = 0 then increase q_{i,j} and q_{i−1,j} by 1 (move I(a))
         if j > 1 and q_{i,j−1} = 0 then increase q_{i,j} and q_{i,j−1} by 1 (move I(b))
         if j > 0 and i < n and q_{i+1,j−1} = 0 then increase q_{i,j−1} and q_{i+1,j−1} by 1 (move I(c))
   endfor
endfor
```

Step 1: **Computing $Trans(P)$ and $Trans(Q)$.**

Lemma 7.2 *After Step 1, $p_{n,0}, q_{n,0} > 0$.*

Proof: Since both P and Q are of degree n, we know that there exists indices i', j', i'', j'' such that $i' + j' = i'' + j'' = n$ and $p_{i',j'}, q_{i'',j''} > 0$. The computation of $P' \in Trans(P)$ increases by one (in the given sequence) the pairs of coefficients $(p_{i',j'-1}, p_{i'+1,j'-1}), (p_{i'+1,j'-2}, p_{i'+2,j'-2}), \ldots, (p_{n-1,0}, p_{n,0})$, and similarly the computation of $Q' \in Trans(Q)$ increases by one (in the given sequence) the pairs of coefficients $(q_{i'',j''-1}, q_{i''+1,j''-1}), (q_{i''+1,j''-2}, q_{i''+2,j''-2}), \ldots, (q_{n-1,0}, q_{n,0})$. Hence, finally, $p_{n,0}, q_{n,0} > 0$. ■

Now, since $\Lambda(P^1) = \Lambda(Q^1)$ (and $p_{n,0}, q_{n,0} > 0$), by using the algorithm of Section 6 for the one-variable case on the polynomials P^1 and Q^1, we can change the coefficients of P^1 such that P^1 becomes identical with the polynomial Q^1. This part of our algorithm (Step 2) is shown below, which clearly takes $O(n)$ time.

After Step 2, we need to transform P^2 such that P^2 becomes identical with Q^2.

for $i = 0, 1, 2, \ldots, n-2$ do
  if $p_{i,0} < q_{i,0}$, increase $p_{i,0}$ and $p_{i+1,0}$ by $q_{i,0} - p_{i,0}$ (move I(a))
  if $p_{i,0} < q_{i,0}$ **and** $p_{i+1,0} > (p_{i,0} - q_{i,0})$, decrease $p_{i,0}$ and $p_{i+1,0}$ by $p_{i,0} - q_{i,0}$ (move II(a))
  if $p_{i,0} < q_{i,0}$ **and** $p_{i+1,0} \leq (p_{i,0} - q_{i,0})$ then
    increase $p_{i+1,0}$ and $p_{i+2,0}$ by $(p_{i,0} - q_{i,0}) - p_{i+1,0} + 1$ (move I(a))
    decrease $p_{i,0}$ and $p_{i+1,0}$ by $p_{i,0} - q_{i,0}$ (move II(a))
endfor

if $p_{n,0} < q_{n,0}$, increase $p_{n-1,0}$ and $p_{n,0}$ by $p_{n,0} - q_{n,0}$ (move I(a))
if $p_{n,0} > q_{n,0}$, decrease $p_{n-1,0}$ and $p_{n,0}$ by $p_{n,0} - q_{n,0}$ (move II(a))

Step 2: **Transforming P^1 to Q^1.**

Let $0 < j_1 \leq n$ be the *largest* index such that there exists some i with $p_{i,j_1} > 0$. For each $0 < j \leq j_1$, let i_j be the largest index such that $p_{i_j,j} > 0$. Since $Support(P') = Support(Q')$ (where $P' \in Trans(P)$ and $Q' \in Trans(Q)$), the following observations hold.

Observation 7.1 *After Step 2, the following statements hold:*

(a) $p_{i,j} = 0$ *if and only if* $q_{i,j} = 0$.

(b) j_1 *is also the largest index such that that there exists some* i *with* $q_{i,j_1} > 0$.

(c) $p_{i,j} = q_{i,j} = 0$ *for all* $j > j_1$ *and any* i.

(d) *For each* $0 < j \leq j_1$, i_j *is also the largest index such that* $q_{i_j,j} > 0$.

(e) $i_{j+1} < i_j$ *for every* $0 < j < j_1$ *(due to applications of move I(c) in Step 1)).*

The following straightforward $O(n^2)$ time algorithm computes the above indices.

find the largest index $j_1 > 0$ such that there exists some i with $p_{i,j_1} > 0$.

for $j = 1, 2, \ldots, j_1$ do
  find the largest index $0 \leq i_j \leq n - j$ such that $p_{i_j,j} > 0$.
endfor

Step 3: **Computing j_1 and indices i_j for $0 < j \leq j_1$**

The remaining part of our algorithm has *at most* j_1 steps (Step 4.1, Step 4.2,. . . ,Step 4.j_1). Consider the polynomials $P_1, P_2, \ldots, P_{j_1}$ and $Q_1, Q_2, \ldots, Q_{j_1}$, where

$$P_j = \sum_{k=0}^{i_j} p_{k,j} x^k y^j, \;\; 1 \leq j \leq j_1$$

$$Q_j = \sum_{k=0}^{i_j} q_{k,j} x^k y^j, \;\; 1 \leq j \leq j_1$$

During Step 4.j ($1 \leq j \leq j_1$), our goal is the following:

Step 4.j(a) If $i_j > 1$, then transform P_j such that $p_{k,j} = q_{k,j}$ for all $0 \leq k \leq i_j - 2$ (i.e., all but *at most two* leading coefficients of P_j are equal to the corresponding coefficients of Q_j) using the algorithm for the one-variable case (via Corollary 6.1). Step 4.j(a) will only change the coefficients of the polynomial P_j (if necessary), and will not change any other coefficient.

Step 4.j(b) Now, if necessary, fix the two remaining coefficients $p_{i_j-1,j}$ and $p_{i_j,j}$ if $i_j > 0$, or fix the remaining coefficient $p_{0,j}$ if $i_j = 0$. Step 4.j(b) will only change, if necessary, the coefficients of P_j and Q_j, and the coefficients of P_{j+1} and Q_{j+1} if $j + 1 \leq j_1$; it will **not change** any other coefficients.

So, a high level description of our Step 4 is as follows.

```
j=1
while (j ≤ j1) do
    execute Step 4.j(a)
    execute Step 4.j(b)
    increase j by 1
endwhile
```

A high level description of Step 4

We first describe Step 4.j(a), assuming $i_j > 1$. Consider applying Corollary 6.1 on the polynomials (in one variable) $\sum_{k=0}^{i_j} p_{k,j} x^k$ and $\sum_{k=0}^{i_j} q_{k,j} x^k$, remembering that it is already true that $p_{k,j}, q_{k,j} > 0$ for $k = 0, 1, \ldots, i_j$. Notice that each valid operation needed for Corollary 6.1, multiplied on both sides by y^j, is also a valid operation for the given instance of the two-variable case which modifies the coefficients such that $p_{k,j} = q_{k,j}$ for all $0 \leq k \leq i_j - 2$. Hence, Step 4.j(a) is as shown in the next page. Clearly, Step 4.j(a) takes $O(i_j) = O(n)$ time and it changes, if necessary, only the coefficients of P_j.

Now, we turn our attention to the details of Step 4.j(b). Remember that $p_{i_j,j}, q_{i_j,j} > 0$. First, we consider a few simple special cases:

Case 1: $i_j = 0$. Then, by Observation 7.1(e), $j = j_1$. Moreover, since $\Lambda(P^2) = \Lambda(Q^2)$, $p_{0,j} = q_{0,j}$ and hence P is identical to Q, completing our transformation and terminating our algorithm.

Case 2: $i_j > 0$ **and** $j = j_1$. Since $\Lambda(P^2) = \Lambda(Q^2)$, $p_{i_j,j} - q_{i_j,j} = p_{i_j-1,j} - q_{i_j-1,j}$. Let $c = p_{i_j,j} - q_{i_j,j}$. If $c < 0$, we increase $p_{i_j,j}$ and $p_{i_j-1,j}$ by $|c|$ (move I(a)). If $c > 0$, we decrease $p_{i_j,j}$ and $p_{i_j-1,j}$ by c (move II(a)). After this, P is identical to Q, completing our transformation and terminating our algorithm.

Case 3: $i_j = 1$. If $j = j_1$, then this case is the same as Case 2. Otherwise, let $j < j_1$. Then, by Observation 7.1(e), $j_1 = j + 1$ and $i_{j_1} = 0$ (hence, $i_{j+1} = i_j - 1$).

if $i_j > 1$ then
  for $k = 0, 1, 2, \ldots, i_j - 2$ do
    if $p_{k,j} < q_{k,j}$, increase $p_{k,j}$ and $p_{k+1,j}$ by $q_{k,j} - p_{k,j}$ (move I(a))
    if $p_{k,j} < q_{k,j}$ **and** $p_{k+1,j} > (p_{k,j} - q_{k,j})$, decrease $p_{k,j}$ and $p_{k+1,j}$ by $p_{k,j} - q_{k,j}$ (move II(a))
    if $p_{k,j} < q_{k,j}$ **and** $p_{k+1,j} \leq (p_{k,j} - q_{k,j})$ then
      increase $p_{k+1,j}$ and $p_{k+2,j}$ by $(p_{k,j} - q_{k,j}) - p_{k+1,j} + 1$ (move I(a))
      decrease $p_{k,j}$ and $p_{k+1,j}$ by $p_{k,j} - q_{k,j}$ (move II(a))
    endif
  endfor
endif

Step 4.j(a)

Let $c_1 = p_{0,j+1} - q_{0,j+1}$, $c_2 = p_{1,j} - q_{1,j}$ and $c_3 = p_{0,j} - q_{0,j}$. Since $\Lambda(P^2) = \Lambda(Q^2)$, $c_1 + c_2 = c_3$. Remember that $p_{0,j+1}, p_{1,j}, p_{0,j}, q_{0,j+1}, q_{1,j}, q_{0,j} > 0$. Our valid operations for this case are as follows:

- If $c_2 > 0$ then increase $p_{0,j+1}$ and $p_{0,j}$ by $|c_2| + 1$ (move I(b)).
- If $c_2 < 0$, then increase $p_{1,j}$ and $p_{0,j}$ by $|c_2|$ (move I(a)). If $c_2 > 0$, then decrease $p_{1,j}$ and $p_{0,j}$ by c_2 (move II(a)).
- Let α and β be the (new) values of $p_{0,j+1} - q_{0,j+1}$ and $p_{0,j} - q_{0,j}$, respectively. $\Lambda(P^2) = \Lambda(Q^2)$ implies that $\alpha = \beta$.
- If $\alpha < 0$, we increase $p_{0,j}$ and $p_{0,j+1}$ by $|\alpha|$ (move I(b)).
- If $\alpha > 0$, we decrease $p_{0,j}$ and $p_{0,j+1}$ by α (move II(b)).

After these modifications, P is identical to Q and the algorithm terminates.

As seen above, the algorithm terminates successfully in either of the above three cases. If either of Case 1, Case 2 or Case 3 does *not* hold, then $i_j > 1$ *and* $j < j_1$. Remember that, by Observation 7.1(e), $i_{j+1} \leq i_j - 1$. There are two more cases to consider as shown below.

Case 4: $i_{j+1} = i_j - 1$. This case is essentially similar to Case 3 (after replacing the indices 0 and 1 by $i_j - 1$ and i_j, respectively), *except* that the algorithm does not terminate at the completion of this case. We repeat the details for the sake of completeness.

Let $c_1 = p_{i_j-1,j+1} - q_{i_j-1,j+1}$, $c_2 = p_{i_j,j} - q_{i_j,j}$ and $c_3 = p_{i_j-1,j} - q_{i_j-1,j}$. Since $\Lambda(P^2) = \Lambda(Q^2)$, $c_1 + c_2 = c_3$. Remember that $p_{i_j-1,j+1}, p_{i_j,j}, p_{i_j-1,j}, q_{i_j-1,j+1}, q_{i_j,j}, q_{i_j-1,j} > 0$. Our valid operations for this case are as follows:

- If $c_2 > 0$ then increase $p_{i_j-1,j+1}$ and $p_{i_j-1,j}$ by $|c_2| + 1$ (move I(b)).
- If $c_2 < 0$, then increase $p_{i_j,j}$ and $p_{i_j-1,j}$ by $|c_2|$ (move I(a)). If $c_2 > 0$, then decrease $p_{i_j,j}$ and $p_{i_j-1,j}$ by c_2 (move II(a)).
- Let α and β be the (new) values of $p_{i_j-1,j+1} - q_{i_j-1,j+1}$ and $p_{i_j-1,j} - q_{i_j-1,j}$, respectively. $\Lambda(P^2) = \Lambda(Q^2)$ implies that $\alpha = \beta$.
- If $\alpha < 0$, we increase $p_{i_j-1,j}$ and $p_{i_j-1,j+1}$ by $|\alpha|$ (move I(b)).

- If $\alpha > 0$, we decrease $p_{i_j-1,j}$ and $p_{i_j-1,j+1}$ by α (move II(b)).

After these modifications, we have modified the coefficients $p_{i_j,j}$ and $p_{i_j-1,j}$ such that they are equal to the coefficients $q_{i_j,j}$ and $q_{i_j-1,j}$, respectively.

Case 5: $i_{j+1} < i_j - 1$. This case is more complicated. Remember that $p_{i_j,j}, q_{i_j,j}, p_{i_{j+1},j+1}, q_{i_{j+1},j+1} > 0$. The valid operation sequence for this case to fix the coefficients $p_{i_j,j}$ and $p_{i_j-1,j}$ are described below.

- Let $c_1 = p_{i_j,j} - q_{i_j,j}$ and $c_2 = p_{i_j-1,j} - q_{i_j-1,j}$. Continue to next step only if $|c_1| + |c_2| > 0$, otherwise this case ends. ($|c_1|+|c_2| = 0$ implies that the two coefficients are already correct).
- Increase $p_{i_{j+1},j+1}$ and $p_{i_{j+1},j}$ by $|c_1|+|c_2|+1$ (move I(b)). This increases the coefficient $p_{i_{j+1},j}$ to a sufficiently large value such that two applications of Lemma 4.1(c) or Lemma 4.1(b), described in the next two steps, can be carried out.
- Since $p_{i_{j+1},j} \geq |c_1| + |c_2| + 1 > |c_1|$, use Lemma 4.1(b) if $c_1 > 0$, or use Lemma 4.1(c) if $c_1 < 0$, to ensure that $p_{i_j,j} = q_{i_j,j}$, at the expense of changing the coefficient $p_{i_{j+1},j}$ *only*. For the sake of completeness, we provide all the details of this step below.
  - Increase, by $|c_1|+1$, the following pairs of coefficients in the given order: $(p_{i_j,j}, p_{i_j-1,j})$, $(p_{i_j-1,j}, p_{i_j-2,j})$, ..., $(p_{i_{j+1}+2,j}, p_{i_{j+1}+1,j})$.
    We need $i_j - i_{j+1} - 1 = O(i_j)$ moves, each of type I(a).
  - If $c_1 < 0$ (resp. $c_1 > 0$), then use alternately increasing/decreasing valid operations in the following manner: increase (resp. decrease) $p_{i_j,j}$ and $p_{i_j-1,j}$, decrease (resp. increase) $p_{i_j-1,j}$ and $p_{i_j-2,j}$, and so forth until we have performed valid operation on the pair $p_{i_{j+1}+1,j}$ and $p_{i_{j+1},j}$.
    We need $i_j - i_{j+1} = O(i_j)$ moves of type I(a) or type II(a).
  - Decrease, by $|c_1|+1$, the following pairs of coefficients in the given order: $(p_{i_{j+1}+1,j}, p_{i_{j+1}+2,j})$, $(p_{i_{j+1}+2,j}, p_{i_{j+1}+3,j})$, ..., $(p_{i_j-1,j}, p_{i_j,j})$. We need $i_j - i_{j+1} - 1 = O(i_j)$ moves, each of type II(a).

  Notice that after these steps, it is still true that $p_{i_{j+1},j} > |c_2|$ (since the total effect was either to increase it, or decrease it by at most $|c_1|$). Also, notice that the coefficient $p_{i_j-1,j}$ does not change its previous value after the completion of this step.
- Since $p_{i_j,j} > |c_2|$, use Lemma 4.1(b) if $c_2 > 0$, or use Lemma 4.1(c) if $c_2 < 0$, to ensure that $p_{i_j-1,j} = q_{i_j-1,j}$, again at the expense of changing the coefficient $p_{i_{j+1},j}$ *only*. For the sake of completeness, we provide all the details of this step below.
  - Increase, by $|c_2|+1$, the following pairs of coefficients in the given order: $(p_{i_j-1,j}, p_{i_j-2,j})$, $(p_{i_j-2,j}, p_{i_j-3,j})$, ..., $(p_{i_{j+1}+2,j}, p_{i_{j+1}+1,j})$.
    We need $i_j - i_{j+1} - 2 = O(i_j)$ moves, each of type I(a).
  - If $c_2 < 0$ (resp. $c_2 > 0$), then use alternately increasing/decreasing valid operations in the following manner: increase (resp. decrease) $p_{i_j-1,j}$ and $p_{i_j-2,j}$, decrease (resp. increase) $p_{i_j-2,j}$ and $p_{i_j-3,j}$, and so forth until we have performed valid operation on the pair $p_{i_{j+1}+1,j}$ and $p_{i_{j+1},j}$.
    We need $i_j - i_{j+1} - 1 = O(i_j)$ moves of type I(a) or type II(a).
  - Decrease, by $|c_2|+1$, the following pairs of coefficients in the given order: $(p_{i_{j+1}+1,j}, p_{i_{j+1}+2,j})$, $(p_{i_{j+1}+2,j}, p_{i_{j+1}+3,j})$, ..., $(p_{i_j-2,j}, p_{i_j-1,j})$. We need $i_j - i_{j+1} - 2 = O(i_j)$ moves, each of type II(a).

  Notice that now both $p_{i_j,j}$ and $p_{i_j-1,j}$ are fixed (that is $p_{i_j,j} = q_{i_j,j}$ and $p_{i_j-1,j} = q_{i_j-1,j}$), at the expense of changing the coefficient $p_{i_{j+1},j}$ only.

- Now, we fix the coefficient $p_{i_{j+1},j}$, if necessary. Let $c_3 = p_{i_{j+1},j} - q_{i_{j+1},j}$. If $c_3 \neq 0$, then we correct the coefficient $p_{i_{j+1},j}$ as follows.
  - If $i_{j+1} = 0$, then we do the following. Since $\Lambda(P^2) = \Lambda(Q^2)$, we must have $p_{0,j+1} - q_{0,j+1} = c_3$. If $c_3 < 0$, then we increase $p_{0,j+1}$ and $p_{0,j}$ by $|c_3|$ (move I(b)). If $c_3 > 0$, then we decrease $p_{0,j+1}$ and $p_{0,j}$ by c_3 (move II(b)). After this, P is identical to Q and the algorithm terminates.
  - If $i_{j+1} > 0$, then we do the following. Increase $p_{i_{j+1},j+1}$ and $p_{i_{j+1}-1,j+1}$ by $|c_3| + 1$ (move I(a)). If $c_3 < 0$, then we increase $p_{i_{j+1},j+1}$ and $p_{i_{j+1},j}$ by $|c_3|$ (move I(b)). If $c_3 > 0$, then we decrease $p_{i_{j+1},j+1}$ and $p_{i_{j+1},j}$ by c_3 (move II(b)). After this, we have corrected the coefficient $p_{i_{j+1},j}$.

  If $c_3 = i_{j+1} = 0$, then $\Lambda(P^2) = \Lambda(Q^2)$ implies that P is identical to Q, hence the algorithm terminates then.

The total time taken in Case 5 is therefore $O(i_j)$.

Hence, the total time taken by Step 4 is at most $O(i_j \cdot j_1) = O(n^2)$ in the unit-cost model. If we start with all numbers with at most B bits, then it is easily seen that during any intermediate step in the algorithm we use numbers with $O(B)$ bits. Hence, we take $O(n^2)$ time in either the unit-cost or the logarithmic-cost model. This completes the proof of Lemma 7.1.

if $i_j = 0$ then algorithm terminates /* Case 1 */

if $i_j > 0$ and $j = j_1$ then /* Case 2 */
  let $c = p_{i_j,j} - q_{i_j,j}$.
  if $c < 0$, we increase $p_{i_j,j}$ and $p_{i_j-1,j}$ by $|c|$ (move I(a)).
  if $c > 0$, we decrease $p_{i_j,j}$ and $p_{i_j-1,j}$ by c (move II(a)).
  algorithm terminates
endif /* end of Case 2 */

if $i_{j+1} = i_j - 1$ then /* Case 3 and Case 4 (combined) */
  let $c_2 = p_{i_j,j} - q_{i_j,j}$
  if $c_2 > 0$, then increase $p_{i_j-1,j+1}$ and $p_{i_j-1,j}$ by $|c_2| + 1$ (move I(b)).
  if $c_2 < 0$, then increase $p_{i_j,j}$ and $p_{i_j-1,j}$ by $|c_2|$ (move I(a)).
  if $c_2 > 0$, then decrease $p_{i_j,j}$ and $p_{i_j-1,j}$ by c_2 (move II(a)).
  let $\alpha = p_{i_j-1,j+1} - q_{i_j-1,j+1}$, $\beta = p_{i_j-1,j} - q_{i_j-1,j}$.
  if $\alpha < 0$, we increase $p_{i_j-1,j}$ and $p_{i_j-1,j+1}$ by $|\alpha|$ (move I(b)).
  if $\alpha > 0$, we decrease $p_{i_j-1,j}$ and $p_{i_j-1,j+1}$ by α (move II(b)).

  if $i_j = 1$ then the algorithm terminates /* terminate if Case 3 */
endif /* end of Case 3 and Case 4 (combined) */

Case 1 to Case 4 of Step 4.j(b)

if $i_{j+1} < i_j - 1$ then /* Case 5 */
  let $c_1 = p_{i_j,j} - q_{i_j,j}$ and $c_2 = p_{i_j-1,j} - q_{i_j-1,j}$

  if $|c_1| + |c_2| > 0$ then

    increase $p_{i_{j+1},j+1}$ and $p_{i_{j+1},j}$ by $|c_1| + |c_2| + 1$ (move I(b)).

    /* First, correct $p_{i_j,j}$ using Lemma 4.1 */
    for $k = i_j, i_j - 1, \ldots, i_{j+1} + 2$ do
      increase $p_{k,j}$ and $p_{k-1,j}$ by $|c_1| + 1$ (move I(a))
    endfor

    for $k = i_j, i_j - 1, i_j - 2, \ldots, i_{j+1} + 1$ do
      if ($c_1 < 0$ and ($i_j - k$ is even)) or ($c_1 > 0$ and ($i_j - k$ is odd)) then
        increase $p_{k,j}$ and $p_{k-1,j}$ by $|c_1|$ (move I(a))
      endif

      if ($c_1 > 0$ and ($i_j - k$ is even)) or ($c_1 < 0$ and ($i_j - k$ is odd)) then
        decrease $p_{k,j}$ and $p_{k-1,j}$ by $|c_1|$ (move II(a))
      endif
    endfor

    for $k = i_{j+1} + 2, i_{j+1} + 1, \ldots, i_j$ do
      decrease $p_{k,j}$ and $p_{k-1,j}$ by $|c_1| + 1$ (move II(a))
    endfor

    /* Next, correct $p_{i_j-1,j}$ using Lemma 4.1 */

    for $k = i_j - 1, i_j - 2, \ldots, i_{j+1} + 2$ do
      increase $p_{k,j}$ and $p_{k-1,j}$ by $|c_2| + 1$ (move I(a))
    endfor

    for $k = i_j - 1, i_j - 2, i_j - 3, \ldots, i_{j+1} + 1$ do
      if ($c_2 < 0$ and ($i_j - 1 - k$ is even)) or ($c_2 > 0$ and ($i_j - 1 - k$ is odd)) then
        increase $p_{k,j}$ and $p_{k-1,j}$ by $|c_2|$ (move I(a))
      endif

      if ($c_2 > 0$ and ($i_j - 1 - k$ is even)) or ($c_2 < 0$ and ($i_j - 1 - k$ is odd)) then
        decrease $p_{k,j}$ and $p_{k-1,j}$ by $|c_2|$ (move II(a))
      endif
    endfor

    for $k = i_{j+1} + 2, i_{j+1} + 1, \ldots, i_j - 1$ do
      decrease $p_{k,j}$ and $p_{k-1,j}$ by $|c_2| + 1$ (move II(a))
    endfor

    /* Now, fix the coefficient $p_{i_{j+1},j}$, if necessary */
    let $c_3 = p_{i_{j+1},j} - q_{i_{j+1},j}$
    if $c_3 \neq 0$ then
      if $i_{j+1} > 0$, then increase $p_{i_{j+1},j+1}$ and $p_{i_{j+1}-1,j+1}$ by $|c_3| + 1$ (move I(a)).
      if $c_3 < 0$, then increase $p_{i_{j+1},j+1}$ and $p_{i_{j+1},j}$ by $|c_3|$ (move I(b)).
      if $c_3 > 0$, then decrease $p_{i_{j+1},j+1}$ and $p_{i_{j+1},j}$ by c_3 (move II(b)).
    endif
    if $i_{j+1} = 0$, then the algorithm terminates.

  endif
endif /* end of Case 5 */

Case 5 of Step 4.j(b)

8 Closing Comments

Looking for efficient algorithms for finding a label representation is itself a most interesting problem for further research, which is not addressed in this paper. Similarly, the full equivalence problem: "are the *equations* of two given systems the same under some change of variables?" is an obvious next question to tackle. We view this paper as merely a first step in the study of a long list of such questions.

References

[1] R. Alur, T.A. Henzinger, and E.D. Sontag, eds., *Hybrid Systems III. Verification and Control*, Springer Verlag, Berlin, 1996.

[2] R. Alur, C. Coucoubetis, N. Halbwachs, T.A. Henzinger, P.H. Ho, X. Nicolin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical Computer Science* **138**(1995): 3-34.

[3] R. Alur and D.L. Dill, "A theory of timed automata," *Theoretical Computer Science* **126**(1994): 183-235.

[4] P.J. Antsaklis, J.A. Stiver, and M. Lemmon, "Hybrid system modeling and autonomous control systems," in *Hybrid Systems* (R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, eds.), (Lec. Notes Comp. Science 736). pp. 366-392, Springer-Verlag, Berlin, 1993

[5] L. Blum, M. Shub and S. Smale, "On a theory of computation and complexity over the real numbers", *Bulletin of the American Mathematical Society*, **21**(1989): 1-46.

[6] S. Eilenberg, and M.P. Schüzenberger, "Rational sets in commutative monoids," *J. Algebra* **13**(1969): 173-191.

[7] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W.H. Freeman, 1979.

[8] S. Ginsburg, and E.H. Spanier, "Bounded Algol-like languages," *Trans. Amer. Math. Soc.* **113**(1964): 333-368.

[9] T.A. Henzinger, "Hybrid automata with finite bisimulations," in *ICALP 95: Automata, Languages, and Programming*, (Z. Fülöp and F. Gécseg, eds) pp. 324-335, Springer-Verlag, Berlin, 1995.

[10] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?," in *Proc. 27th Annual Symposium on Theory of Computing* pp. 373-382, ACM Press, 1995.

[11] G. Lafferriere, G.J. Pappas, and S. Sastry, "Minimal hybrid systems," submitted.

[12] O. Maler, ed., *Hybrid and Real-Time Systems*, Springer-Verlag, Berlin, 1997.

[13] A. Puri and P. Varaiya, "Decidability of hybrid systems with rectangular differential inclusions," in *Computer Aided Verification* (D.L. Dill, ed.) (Lec. Notes Comp. Science 818). pp. 95-104, Springer-Verlag, Berlin, 1994.

[14] A. Puri and P. Varaiya, "Decidable hybrid systems," *Math. Comput. Modelling* **23**(1996): 191-202.

[15] J-P. Quadrat, "Max-plus algebra and applications to system theory and optimal control," in *Proceedings of the International Congress of Mathematicians*, (Zürich, 1994), Birkhäuser, Basel, 1995, pp. 1511–1522,

[16] E.D. Sontag, "Nonlinear regulation: The piecewise linear approach," *IEEE Trans. Autom. Control* **AC-26**(1981): 346-358.

[17] E.D. Sontag, "Remarks on piecewise-linear algebra," *Pacific J.Math.*, **98**(1982): 183-201.

[18] E.D. Sontag, "Real addition and the polynomial hierarchy," *Inform. Proc. Letters* **20**(1985): 115-120.

[19] E.D. Sontag, *Mathematical Control Theory: Deterministic Finite Dimensional Systems. Second Edition*, Springer, New York, 1998.

[20] E.H. Spanier, *Algebraic Topology*, McGraw-Hill, New York, 1966.